
Les listes chainées

Langage C
BT2

Anthony Desvernois
v. 1.0
9 pages
BT2-PROGC-TP05



Propriétés du document

Auteur	Anthony Desvernois
Validé par	Julien Stuyck
Version	1.0
Nombre de pages	9
Référence	BT2-PROGC-TP05

Historique du document

Date	Version	Auteur	Résumé des modifications
12/02/2009	1.0	Anthony Desvernois	Version initiale
15/02/2009	1.1	Anthony Desvernois	Revue du sujet

Accès au document

La dernière version de ce document est téléchargeable à l'adresse suivante :
www.intranet.isbp.fr

Destinataires

Ce document est destiné aux étudiants de Sup'Biotech en BT2

Contents

1. But du TP	4
2. Les bases.....	5
2.1. Création d'un étudiant	5
2.2. Ajout en tête.....	5
2.3. Ajout en queue	5
2.4. Recherche.....	6
3. Un peu de manipulation.....	7
3.1. Suppression	7
3.2. Insertion à une position n	7
3.3. Echange	8
4. Warrior	9
4.1. Insertion en place	9
4.2. Tri d'une liste.....	9

1. But du TP

Le but du TP est de vous entraîner à la manipulation des listes chaînées vu en TD. Pour cela, vous allez utiliser une liste d'élève identifié par leur UID.

La structure que vous utiliserez pour manipuler les étudiants est la suivante :

```
typedef struct student {
    unsigned int uid;
    struct student* next;
} s_student;
```

Une archive contenant les sources du TP (à compléter) est disponible à l'adresse suivante :

http://perso.isbp.fr/~desver_a/isbp/bt2-progc-tp05.tar.bz2

Pour rappel, l'extraction de cette archive se fera via la commande suivante :

```
$ tar xjf bt2-progc-tp05.tar.bz2
```

Vous trouverez dans cette archive les fichiers suivants :

- main.c : il contient la fonction principale de ce TP avec un ensemble de tests basiques
- list.h : il contient la définition de notre structure s_student
- bases.c
- medium.c
- warrior.c

L'ensemble des fichiers fournis et compilable, en conséquence certaines fonctions sont déjà pré-rempli avec une instruction return. N'oubliez pas de modifier cette instruction lorsque vous implémenterez la fonction.

2. Les bases

2.1. Création d'un étudiant

But	Allouer une structure seule contenant l'UID d'un étudiant
Fichier	bases.c

Prototype :

```
s_student* create_student(unsigned int uid);
```

2.2. Ajout en tête

But	Ajout d'un élève à une liste d'étudiants en tête
Fichier	bases.c

Prototype :

```
s_student* insert_head(unsigned int uid, s_student* list);
```

2.3. Ajout en queue

But	Ajout d'un élève à une liste d'étudiants en queue
Fichier	bases.c

Prototype :

```
s_student* insert_tail(unsigned int uid, s_student* list);
```

2.4. Recherche

But	Recherche d'un élève dans une liste chaînée
Fichier	bases.c

Prototype :

```
int find(unsigned int uid, s_student* list);
```

La fonction doit retourner 1 si l'élève est présent dans la liste, 0 sinon.

3. Un peu de manipulation...

3.1. Suppression

But	Suppression d'un élève dans une liste chaînée
Fichier	bases.c

Prototype :

```
s_student* delete(unsigned int uid, s_student* list);
```

Si l'élève n'existe pas, ne rien faire.

3.2. Insertion à une position n

But	Insertion dans une liste à la place n (index à 0)
Fichier	medium.c

Prototype :

```
s_student* insert_n(unsigned int uid, unsigned int place,  
                    s_student* list);
```

Si la place n'est pas valide, insérer en queue.

3.3. Echange

But	Echangé la place de deux étudiants dans une liste chaînée
Fichier	medium.c

Prototype :

```
s_student* swap(unsigned int uid, unsigned int uid2,  
                s_student* list);
```

Si l'un des deux étudiants n'existe pas, ne rien faire. Vous devez effectuer un échange des structures, et pas uniquement des UID.

4. Warrior

4.1. Insertion en place

But	Insérer un étudiant dans une liste chaînée triée (ordre croissant)
Fichier	warrior.c

Prototype :

```
s_student* insert(unsigned int uid, s_student* sorted_list);
```

Vous pouvez considérer que la liste passée en argument est triée. Le tri est fait en fonction des UID.

4.2. Tri d'une liste

But	Trié la liste donnée en argument (ordre croissant)
Fichier	warrior.c

Prototype :

```
s_student* sort(s_student* unsorted_list);
```

Le tri doit être fait en fonction des UID.